

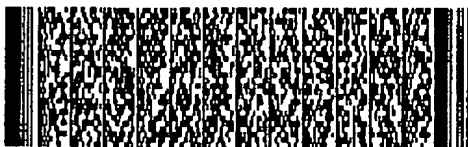
申請日期: 88.12.21	案號: 88122503
類別: G06F 11/00, G06F 12/00, G06F 9/445	

(以上各欄由本局填註)

**公告本****發明專利說明書**

449685

一、 發明名稱	中 文	局部缺陷記憶體的處理方法和系統
	英 文	
二、 發明人	姓 名 (中文)	1. 林錫聰
	姓 名 (英文)	1.
	國 籍	1. 中華民國
	住、居所	1. 台北市大同區朝陽里15鄰延平北路二段69號九樓
三、 申請人	姓 名 (名稱) (中文)	1. 華邦電子股份有限公司
	姓 名 (名稱) (英文)	1.
	國 籍	1. 中華民國
	住、居所 (事務所)	1. 新竹科學工業園區研新三路四號
	代表人 姓 名 (中文)	1. 焦佑鈞
	代表人 姓 名 (英文)	1.



## 四、中文發明摘要 (發明之名稱：局部缺陷記憶體的處理方法和系統)

一種局部缺陷記憶體的處理方法，可以適用在包含一缺陷記憶單元的記憶體和待載入此記憶體的原始程式碼之間。首先，掃描此原始程式碼，並且在缺陷記憶單元所對應的缺陷位址前後，決定出第一分斷點和第二分斷點。接著移動第一分斷點和第二分斷點之間的區段程式碼到程式碼之不對應缺陷位址的第一位址和第二位址之間。接著，連接移動後之區段程式碼與原始程式碼中未移動部分的執行順序，並且調整移動後的區段程式碼與原始程式碼中未移動部分之間或是區段程式碼本身的參考位址。最後，便可以將更改過之程式(含原始程式碼中未移動部分、連接指令以及移動後的區段程式碼)載入至記憶體中。因此，所載入的更改過之程式碼在執行時，可以避開缺陷記憶單

## 英文發明摘要 (發明之名稱：)



四、中文發明摘要 (發明之名稱：局部缺陷記憶體的处理方法和系統)

元且不影響原程式碼之執行功能。

英文發明摘要 (發明之名稱：)



449685

本案已向

國(地區)申請專利

申請日期

案號

主張優先權

無

有關微生物已寄存於

寄存日期

寄存號碼

無

## 五、發明說明 (1)

本發明係有關於一種局部缺陷記憶體的處理方法和系統，特別提出一種處理方法和系統，能夠針對尚未載入記憶體或是已存在於記憶體內的程式碼加以調整，而使得此程式碼仍可以載入至具有局部缺陷記憶單元的記憶體內並且加以執行。

以往如果記憶體IC內存在任何缺陷記憶單元(defective memory cell)時，一般是不會將這樣的記憶體IC在市面上販售。為了減少因單一或少數缺陷而廢棄整個IC的情況，目前有許多技術可以讓局部缺陷的記憶體IC，在操作上宛如完全正常的記憶體IC。

例如，美國專利NO. 4939694中即揭露一種能夠自我測試(self-testing)和自我修補(self-repairing)的記憶體系統。此記憶體系統可以在使用現場進行自我測試，以便定位出其中的缺陷記憶單元。一旦發現任何缺陷記憶單元，此記憶體系統便會使用一種稱為錯誤修正碼引擎(error correction code engine)的裝置，對於這些缺陷記憶單元進行修補。如果錯誤修正碼引擎無法負荷，則記憶體系統便會取代掉這些缺陷記憶單元。

另外在美國專利No. 5644541中則是利用替換記憶體(substitution memory)來處理含有缺陷記憶單元的半導體記憶體。當半導體記憶體內存在數個已知位置的錯誤位元時，則利用一映射邏輯(mapping logic)將所有需要存取到這些錯誤位元的存取請求，導向到替換記憶體內良好的記憶單元，藉此取代原本的缺陷記憶單元。



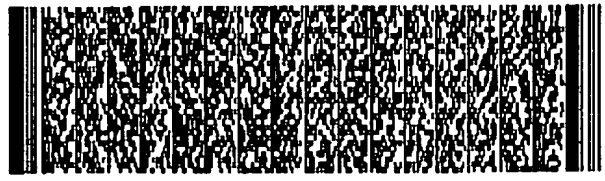
## 五、發明說明 (2)

美國專利No. 5278847 中則是揭露一種使用錯誤偵測/修正碼(error detecting and correcting code, EDAC)的容錯(fault-tolerating)記憶系統。儲存資料的可靠度可以藉由對於每個資料字元加入EDAC編碼和增加備用位元(spare-bit)而達成。

美國專利No. 5579266 中則是利用雷射修補技術和可程化熔絲修補技術來處理缺陷記憶單元，可以利用冗餘記憶體來取代缺陷記憶體。

其中像是美國專利NO. 4939694 中所揭露的錯誤檢測和修補技術、美國專利No. 5278847 中所揭露的編碼和備用位元技術，亦或是美國專利No. 5644541 中所揭露的存取導向技術，不僅會增加硬體設計上的困難度，同時在可執行機器碼每次執行時也會造成程式執行上的負擔。另外，如美國專利No. 5579266 所揭露的冗餘記憶體技術，在硬體設計上或是在實際進行修補的程序上也都十分複雜。

另一方面，一般可執行的機器程式碼是由電腦程式的原始碼(source code)，利用編輯器(compiler)和連結器(linker)所產生。機器程式碼則可以直接載入記憶體中來執行。第1圖表示一般單晶片電腦(single chip computer)或單晶片系統(system-on-a-chip)之系統架構圖。如圖所示，此系統包括CPU(central processing unit，或稱微處理器)1、RAM(random access memory，隨機存取記憶體)3、ROM(read-only memory，唯讀記憶體)5、I/O界面7以及外部儲存裝置9，例如硬碟機、軟碟

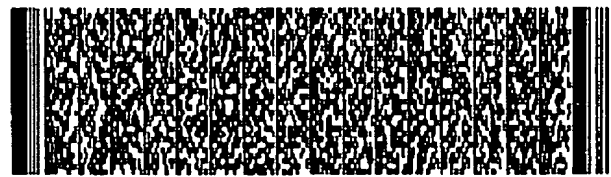


## 五、發明說明 (3)

機或CD-ROM等等。在正常操作情況下，CPU 1會透過資料/位址匯流排10，由I/O界面7或是外部儲存裝置9將預備執行的機器程式碼，載入到RAM 3中。一般機器程式碼中包含了三個部分，分別為指令(instruction)、資料(data)和堆疊(stack)。CPU 1會從指令部分的程式進入點(entry point)開始執行此機器程式碼。

第2圖表示一般機器程式碼在執行(execution)時的流程圖。首先，取得程式進入點(S1)。接著取得下一個指令的位址(address)(S2)，並且根據此位址，讀取下一個指令中的運算子(opcode)(S3)。接著將運算子進行解碼(S4)，並且根據指令形式來判斷是否需要運算元(operand)。如果此指令需要運算元，則再從後續位址中讀取運算元(S5)。最後根據運算子所代表的指令動作以及運算元所代表的資料內容或參考位址，執行該指令(S6)。如果此指令為程式終止指令(S7)，則機器程式碼執行完成(S8)，否則回到步驟S2取得下一個指令的位址。必須說明的是，每個指令並不一定包含相同的位元組長度，這與所採用的CPU類型有關。一般CPU所採用的指令集可分為可變長度碼字指令集和固定長度碼字指令集。

在上述執行流程中，主要是針對機器程式碼中的指令部分。如果相同的執行程序應用於資料部分或是堆疊部分，其解碼結果則會完全錯亂。當某個資料位元組由CPU 1解碼成某個錯誤的運算子後，則會根據錯誤的運算子讓後面數個資料被誤認為運算元。一般CPU 1是無法透過標

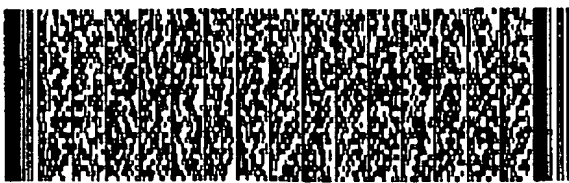


## 五、發明說明 (4)

準的擷取和解碼動作來辨認出何者為指令部分，何者為資料部分或堆疊部分。另外，程式碼的指令部分和資料/堆疊部分亦不可以任意地加以分斷，指令部分的分斷必須根據各指令的格式而定，亦即各指令(包含運算元和運算子)的位元組長度。而資料/堆疊部分一般是無法分斷的，這是因此在資料/堆疊部分中可能包含只有在執行時才能判斷的資料關連性，例如資料結構中的陣列。

有鑑於此，本發明的主要目的，在於提供一種局部缺陷記憶體的处理方法和系統，能夠在不改變硬體架構和不增加軟體執行負擔的前提下，分別可以在程式碼未載入或是程式碼已存在的情況中，修改程式碼在此局部缺陷記憶體中的儲存方式，以避免使用到缺陷記憶單元。

根據上述之目的，本發明提出一種局部缺陷記憶體的處理方法，可以適用在包含複數個記憶單元的記憶體和待載入此記憶體的原始程式碼之間。假設在此記憶體中具有至少一缺陷記憶單元。首先，掃描此原始程式碼，並且在缺陷記憶單元於原始程式碼中所對應的缺陷位址前後，決定出第一分斷點和第二分斷點。接著移動第一分斷點和第二分斷點之間的區段程式碼到第一位址和第二位址之間，其中第一位址和第二位址之間不包含上述缺陷位址。當被移動的區段程式碼中包含至少一可執行之指令時(亦即包含指令部分)，則需要連接移動後之區段程式碼與原始程式碼中未移動部分的執行順序。而當移動後之區段程式碼與原始程式碼中未移動部分之間存在參考位址或是區段程



## 五、發明說明 (5)

式碼本身的內部存在參考位址，則視情況進行調整。最後，便可以將原始程式碼中未移動部分、連接指令以及移動後的區段程式碼依序載入至記憶體中。由於利用上述方式所載入的程式碼為可執行狀態並且不會儲存於已知的缺陷記憶單元之中，因此即使記憶體中不存在缺陷記憶單元，此記憶體仍可以正常地使用。另外，上述處理過程中不涉及硬體電路的修改或變更，因此實施成本相當低。

另外，如果待載入的原始程式碼本身已經過模組化處理，則在進行程式碼掃描時便可以直接決定所需要的分斷點。但是如果原始程式碼並未預先加以模組化處理，則依序讀取原始程式碼，再依據原始程式碼中各指令組成，輸出複數可分斷點，最後便可以根據缺陷位址和待插入的連接指令長度，決定出所需要的第一分斷點和第二分斷點。要確定掃描到完整的原始程式碼，本發明則提供下列方法。首先，提供第一資料表和第二資料表，其中第一資料表用來記錄原始程式碼中條件分支指令的分支目的位址，而第二資料表則用來記錄已讀取的位址範圍。當所讀取的指令為一條條件分支指令時，便在第一資料表中記錄下條件分支指令之分支目的位址；當完成一指令的讀取時，則更新第二資料表的位址範圍。當所讀取的指令為一結束指令或是其位址在第二資料表之位址範圍內的時，則只要第一資料表的分支目的位址不屬於第二資料表的位址範圍內，便可以根據第一資料表的分支目的位址繼續讀取。透過上述處理分支和迴圈的方式，便可以確保掃描到所有的指令。

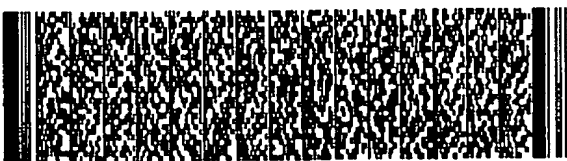


## 五、發明說明 (6)

另外，要連接移動後的區段程式碼和原始程式碼中未移動部分，可以插入兩個無條件分支指令來完成。第一個無條件分支指令插入在第一分斷點的位址上，其目的位址為區段程式碼移動後之第一位址；第二個無條件分支指令插入在區段程式碼移動後的第二位址，其目的位址則為第二分斷點。

上述處理方式在實用上亦可以略加變更。緣此，本發明另外提出了一種局部缺陷記憶體處理方法，其程序與上述方式有些許不同。首先，掃描原始程式碼並且在缺陷記憶體單元對應的缺陷位址前後，決定第一分斷點和第二分斷點。接著，先將原始程式碼整個載入到記憶體中，而另外在不包含缺陷記憶體單元的第一記憶單元和第二記憶單元之間，再次載入位於第一分斷點和第二分斷點之間的區段程式碼。接著則與前述方式相同：連接再載入之區段程式碼與原始程式碼中其他部分的執行順序、修正彼此間的參考位址。透過此方式，除了可以達到前一方式的目的外，由於程式碼先被載入到記憶體中，還可以加快後續處理步驟的速度，例如修正參考位址的處理。

另外，本發明另提供一種局部缺陷記憶體的處理方法，可以適用於原始程式碼已存在於記憶體中的情況。首先，檢查記憶體並且找出記憶體中功能弱化之至少一缺陷記憶單元。接著掃描原始程式碼，以便在缺陷記憶單元所對應的缺陷位址前後，決定出第一分斷點和第二分斷點。接著將第一分斷點和第二分斷點之間的區段程式碼，移動



## 五、發明說明 (7)

到至記憶體中不包含上述缺陷記憶單元的第一記憶單元和第二記憶單元之間。接著則與前述方式相同：連接移動後之區段程式碼與原始程式碼中未移動部分的執行順序、修正彼此間的參考位址。由於此方式可以適用於使用中的記憶體，因此在應用上更為方便。

## 圖式之簡單說明：

為使本發明之上述目的、特徵和優點能更明顯易懂，下文特舉一較佳實施例，並配合所附圖式，作詳細說明如下：

第1圖表示一般單晶片電腦(single chip computer)或單晶片系統(system-on-a-chip)之系統架構圖。

第2圖表示一般機器程式碼在執行(execution)時的流程圖。

第3圖表示本發明第一實施例之局部缺陷記憶體處理系統的架構圖。

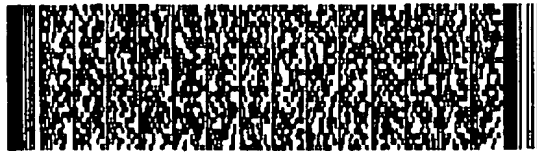
第4圖表示本發明第一實施例之局部缺陷記憶體處理方法的流程圖。

第5圖表示在本發明中程式碼掃描方法的流程圖。

第6圖表示根據本發明之程式碼掃描方法處理一程式碼範例的示意圖。

第7圖表示在本發明中連接移動後區段程式碼和原始程式碼未移動部分的示意圖。

第8圖表示在本發明中修正參考位址的示意圖。



## 五、發明說明 (8)

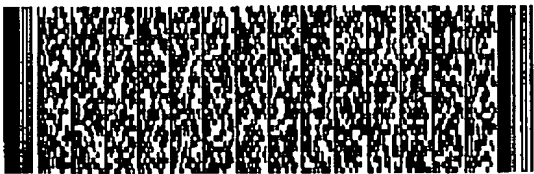
第9圖表示本發明第二實施例之局部缺陷記憶體處理方法的流程圖。

第10圖表示本發明第三實施例之局部缺陷記憶體處理系統的架構圖。

第11圖表示本發明第三實施例之局部缺陷記憶體處理方法的流程圖。

## 符號說明：

- 1~CPU；
- 3~RAM(記憶體)；
- 5~ROM；
- 7~I/O界面；
- 9~外部儲存裝置；
- 10~資料/位址匯流排；
- 20~原始程式碼；
- 30~第一資料表；
- 40~第二資料表；
- C1-C7~指令段部分；
- D1-D3~資料/堆疊部分；
- 21~缺陷位址；
- 51、52~連接指令；
- 50~移動區段程式碼後的程式碼；
- 60~完成參考位址修正的程式碼；
- R1-R3、R1'-R3'~參考位址。



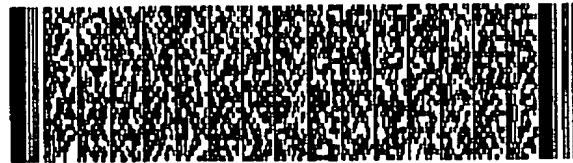
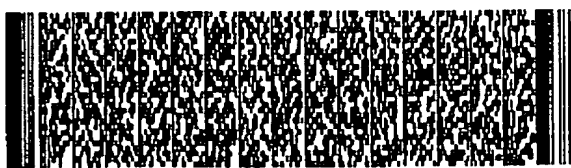
## 五、發明說明 (9)

## 實施例：

本發明的局部缺陷記憶體處理方法和系統，主要是修正待載入或已存在的原始程式碼，以避開發生缺陷的記憶體單元，藉此讓具有局部缺陷記憶體單元的記憶體IC仍可以正常運作。修正原始程式碼的處理方式，主要是透過辨識機器程式碼的可分斷點(break point)和移動區段程式碼的方式來達成。亦即，先辨識出原始程式碼中的可分斷點，再根據實際發生缺陷的記憶體單元所對應到的程式位置，決定出可以移動的區段程式碼；將此區段程式碼移動到可以正常工作的記憶體單元後，再利用連接指令，維持移動後的區段程式碼和原始程式碼之間的執行順序，便可以避開使用到缺陷記憶體單元。由於本發明並不需要變更或重新設計硬體電路，並且在軟體執行上所增加的額外負擔也非常有限(僅增加連接指令)，因此可以達到本發明之目的。以下配合圖式，詳細說明本發明各實施例之技術內容。

## 第一實施例：

第3圖表示第一實施例之局部缺陷記憶體處理系統的架構圖。在第3圖中，記憶體3包含至少一個缺陷記憶體單元，而在以下的描述中，除了特別指出外，均以單一缺陷記憶體單元或相鄰之一組缺陷記憶體單元為預設條件。CPU 1則是預備透過資料/位址匯流排10將原始程式碼20載入到記憶體3，而原始程式碼20原本預備載入的記憶體單元中就



## 五、發明說明 (10)

包括上述的缺陷記憶單元。因此，在CPU 1中所執行的載入器(loader)，必須執行一些前置處理動作，才能夠保證載入後的原始程式碼20不會佔用到缺陷記憶單元，同時原始程式碼20本身仍可以正常的執行。在本實施例中，CPU 1和記憶體3可以共存於同一積體電路中，例如單晶片系統(system on a chip)的應用，另外，亦可以是個別獨立的積體電路，此時CPU 1與記憶體3透過特定的IC接腳相連接，用來執行後續所描述之缺陷檢查、處理等等步驟。

第4圖表示第一實施例之局部缺陷記憶體處理方法的流程圖，其中詳述CPU 1在載入原始程式碼20前所必須執行的各種步驟。首先，CPU 1必須決定在記憶體3中缺陷記憶單元的位置(S10)。亦即，需要對於記憶體3進行缺陷測試，以便檢查出其中是否包含有缺陷記憶單元以及其實體位置。一種檢查記憶單元是否為缺陷的例子，是將資料"1"和"0"寫入每個記憶單元，再讀出其資料是否正確，如果寫入資料和讀出資料不吻合，即表示該記憶單元有缺陷。實際執行測試的裝置可以是CPU 1本身，亦可以由外部的測試器或電腦來執行。

根據經測試所得到的缺陷記憶單元實體位址以及原始程式碼20的載入訊息(例如載入開始點位址)，便可以決定出此缺陷記憶單元在原始程式碼20中所對應的缺陷位址(S11)，亦即此缺陷記憶單元在原始程式碼20所代表的絕對參考位址。由於缺陷記憶單元無法正常操作，因此CPU 1所執行的載入器必須讓原本預定儲存於此缺陷記憶單元

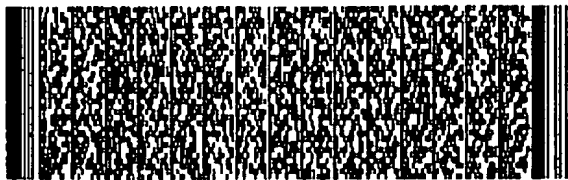


## 五、發明說明 (11)

的程式碼部分，變更載入到其他的記憶單元，以避免資料的流失。然而，根據前述對於一般機器程式碼的描述可知，機器程式碼是不可以任意分斷的，某一個位元組可能僅是某個指令的一部分，不可以單獨加以移動，以避免指令和資料的錯亂。因此，CPU 1 必須在原始程式碼20 中找出適當的分斷點，才能正確地移動原本佔用到缺陷記憶單元的位元組。

接著，對於原始程式碼20 進行掃描(S12)，以便找出原始程式碼20 中的所有可分斷點。在一般機器程式碼中，指令部分中的個別指令間均為可分斷點，這是因為即使個別指令被分斷，在執行時CPU 1 仍可以正確解碼出正確的指令訊息；相對地，資料部分和堆疊部分則不可分斷，這是因為資料和堆疊之間可能具有在執行時才可能檢查出來的相關性，例如陣列資料。另一方面，CPU 1 無法直接分辨出機器程式碼中的指令部分和其他部分。因此，CPU 1 必須對原始程式碼20 中的指令部分整個進行掃描，才可以決定出所有的可分斷點。

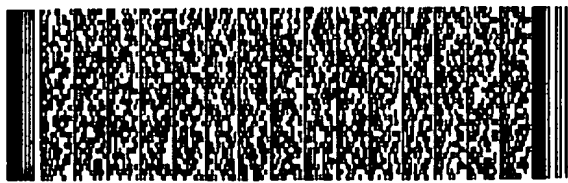
第5圖表示本實施例中原始程式碼20 的掃描方法流程圖。此處所謂的"掃描"，是包含擷取運算子、解碼、擷取運算元這些動作，但是不需要執行。為了掃描原始程式碼20 的整個指令部分，必須特別處理一般機器程式碼中常見的兩種特殊型態，即分支(branch)和迴圈(loop)。分支型態會在原始程式碼20 中產生兩種不同的指令執行順序，一般是由條件分支指令(conditional branch



## 五、發明說明 (12)

instruction) 所造成，例如 JNE、JE、JG 等等。在掃描過程中，為了確定處理到所有的指令，因此兩種指令執行順序都需要加以掃描。在本實施例中，掃描過程中會建立第一資料表 30 (如第 5 圖所示)，當遇到任何條件分支指令時，會將該分支指令之下一位址或該分支指令之分支目的位址 (branch-to address) 儲存於第一資料表 30 中。藉此，當主要指令執行順序完成掃描後，便可以根據第一資料表 30 進行其他執行順序的掃描。另外需要說明的是，一般分支指令還包含一種無條件分支指令 (unconditional branch instruction)，例如 JMP，但是由於這種指令並不會造成兩條不同的指令執行順序，因此本實施例中之掃描可依其目的位址 (branch-to address) 繼續掃描。另一方面，迴圈型態則會造成掃描無法終止。而在本實施例中，則是在掃描過程中建立第二資料表 40，用來記錄已經被掃描過的位址範圍，而每掃描一個指令，即會更新第二資料表 40。當重複掃描到迴圈內部時，便可以根據第二資料表 40 判斷出已掃描過此部分，因此可以防止繼續掃描後續的指令。

以下詳細說明第 5 圖中各步驟的動作。首先取得程式進入點 (S110)。接著取得並指定下一個指令的位址 (S111)，並且根據此位址，讀取下一個指令中的運算子並且進行解碼 (S112)。如果此指令需要運算元，則再從後續位址中讀取運算元 (S113)。此時所讀取者為一完整指令，因此可以記錄為一可分斷點。接著判斷此指令是否為條件



## 五、發明說明 (13)

分支指令(S114)，如果是，則可：(i)將此條件分支指令的分支目的位址加入到第一資料表30中(S115)，而依該條件分支指令之下一位址繼續掃描；或(ii)將該條件分支指令的下一位址加入到第一資料表中，而依該條件分支指令之分支目的位址繼續掃描。接著決定出掃描之下一個位址(S116)，判斷此位址的指令是否為返回指令(例如RET)或程式結束指令(例如END)(S117)。如果下一個位址的指令不是返回指令或是程式結束指令，則根據第二資料表40來判斷此一位址是否為已掃描位址範圍(S121)。如果此下一個位址亦非已掃描位址範圍，則修改第二資料表40以更新已掃描位址範圍之後(S122)，繼續處理下一個指令。如果在步驟S117中下一個位址的指令的確是返回指令或是程式結束指令，亦或在步驟S121中下一個位址為已掃描過的位址範圍，則判斷第一資料表30中是否仍有待掃描位址(S118)。如果仍有，則從第一資料表30中取得一待掃描位址並且將其從第一資料表30中移去(S119)，再修改第二資料表40中的已掃描位址範圍(S122)，繼續處理此一待掃描位址的後續指令。如果第一資料表30中已經沒有待掃描位址時，則表示已經掃描完成全部的指令(S120)。此時已經得到所有的可分斷點，並且可以判斷出原始程式碼20中的指令部分和資料/堆疊部分。在第5圖中，虛線框P1表示處理分支型態的相關步驟，虛線框P2則表示處理迴圈型態的相關步驟。

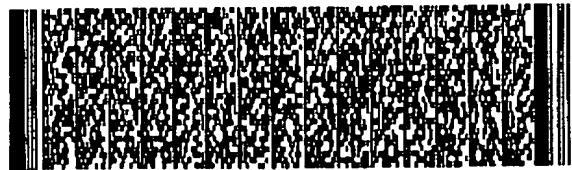
上述掃描過程中的處理動作，可以簡要描述如下：



## 五、發明說明 (14)

(1) 當所讀取的指令為一條條件分支指令(例如JNE、JE、JG等等)時，將條件分支指令之分支目標位址或下一位址記錄於第一資料表30中；(2) 當完成讀取一指令時，便更新第二資料表40之位址範圍並且輸出可分斷點；(3) 如果所讀取的指令為一結束指令或返回指令或者是其位址在第二資料表40的位址範圍內，只要第一資料表30的某個待掃描位址不屬於第二資料表40的位址範圍，便從第一資料表30的這個待掃描位址繼續讀取。

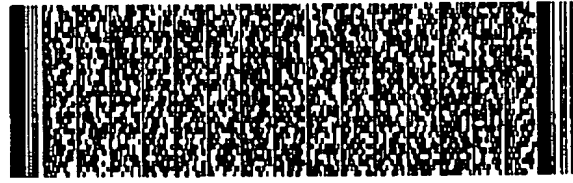
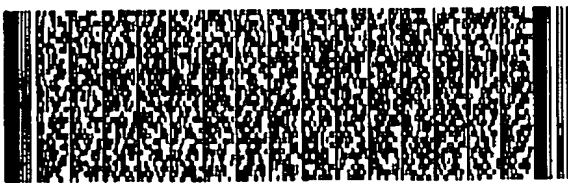
根據第5圖所述的掃描程序，以下以一實例來說明其掃描的動作。第6圖表示根據第5圖掃描方法來處理一程式碼範例的示意圖。其中，程式進入點為Q1。當從Q1掃描至Q2，讀取到一個條件分支指令JNE，此時在第一資料表30中記錄此條件分支指令JNE的分支目標位址Q6。當繼續掃描至Q3時，則讀取到一個非條件分支指令JMP。對於非條件分支指令JMP不需要儲存其分支目標位址Q10，而是直接到其分支目標位址繼續進行掃描。接著從Q10掃描到Q11，則依序讀取到兩個條件分支指令JG和JE。同樣的，在第一資料表30則記錄這兩個指令的分支目標位址Q4和Q8後，繼續進行掃描。當掃描到Q12時，讀取到程式結束指令END，因此結束主程式段的掃描。此時，已掃描的區域包括C1、C2、C6、C7，而第一資料表30中則記錄Q6、Q4和Q8的待掃描位址。接著，從第一個待掃描位址Q6繼續進行掃描，直到Q7讀取到返回指令RET為止。接著再從第二個待掃描位址Q4繼續進行掃描，直到掃描到Q5，雖然並未讀取到任何



## 五、發明說明 (15)

結束指令或是返回指令，不過其下一個位址Q6係屬於已掃描區域，因此仍結束掃描。接著再從第三個待掃描位址Q8繼續進行掃描，直到掃描到Q9。同樣的，雖然並未讀取到任何結束指令或是返回指令，不過其下一個位址Q10係屬於已掃描區域，因此仍結束掃描。至此，第一資料表30中已經沒有其他的待掃描位址，因此整個掃描工作完成。在掃描過程中，可以判斷出所有的可分斷點（即每一完整的指令之前後皆為可分斷點），而且判斷出C1-C7係屬於指令部分，而D1-D3則屬於資料或堆疊等非指令部分，並且當掃描完成但仍未被掃描到之一連續程式碼區段，可被視為一資料或堆疊等非指令區段。

回到第4圖，當完成原始程式碼20的掃描動作，便可以根據所得到的可分斷點，在缺陷記憶單元所對應的缺陷位址前後分別決定第一分斷點和第二分斷點(S13)。每一完整的指令之間或之前之後，皆為可分斷點。第一分斷點和第二分斷點之間的指令部分（也可能包含資料/堆疊部分）稱為區段程式碼，而區段程式碼需要被移動到其他的位址上，以避開缺陷位址。在本實施例中，第一分斷點和第二分斷點的決定方式，是選擇以最接近缺陷位址但是與缺陷位址之間仍存在數個位元組的分斷點為準，亦即第一/第二分斷點與缺陷位址之間仍存在數個良好的記憶單元，可以用以儲存連接指令（此點稍後詳述）。在以下的描述中，均以此種情況為例進行說明。不過上述的分斷點選擇方式並非用以限定本發明。舉例來說，第一/第二分斷



## 五、發明說明 (16)

點也可以根據與缺陷位址間至少間隔數個可分斷點的方式來選擇。另外，如果後續的移動方式是採用將缺陷位址後的所有程式碼往下位移一定長度位元組的方式，此時第一分斷點可以根據在缺陷位址之前且與缺陷位址之間仍存在數個位元組的方式來選擇，而第二分斷點則設在程式碼的最後，即該程式碼最後一個位元組之後，而將缺陷位址附近及後方的程式碼設為需要移動的區段程式碼，而將該區段程式碼移至該缺陷位址之後，仍可以達到本發明之目的。

決定第一分斷點和第二分斷點之間的區段程式碼之後，接著則可以將此區段程式碼移動到第一位址和第二位址之間(S14)。必須注意的是，缺陷位址不應該位於第一位址和第二位址之間，也就是區段程式碼在載入後的記憶單元中不包含有缺陷記憶單元。在完成區段程式碼的重新定址後，如果被移動的區段程式碼包含指令部分，接著必須插入連接指令，讓原始程式碼仍維持其程式執行順序。在本實施例中，主要是插入兩個無條件分支指令(JMP)來達成連接的目的。第一個無條件分支指令係插入於第一分斷點的位址上，其分支目的位址是指向移動後的區段程式碼；第二個無條件分支指令係插入於移動後區段程式碼的後面，而其分支目的位址則是指向第二分斷點的位址，但如果第二分斷點之位址係設在程式碼的最後，則不需要該第二個無條件分支指令。因此，當程式碼執行到第一分斷點位址上的指令(其為區段程式碼之開始部分)時，則會透

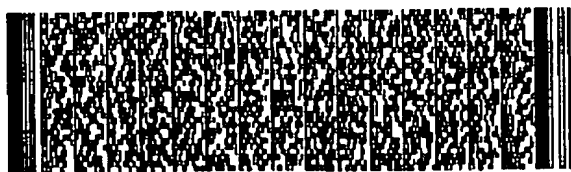


## 五、發明說明 (17)

過第一個無條件分支指令，跳到移動後的區段程式碼繼續執行；當完成移動後區段程式碼的指令後，則透過第二個無條件分支指令跳回到第二分斷點位址上的指令。另外，如果區段程式碼只是單純的資料/堆疊部分，則不需要加入連接指令，只需要修改與其相關的參考位址即可。

第7圖表示本實施例中連接移動後區段程式碼和原始程式碼未移動部分的示意圖。如圖所示，原始程式碼20包含了各程式碼A-G，而缺陷位址21則位於程式碼E內，在程式碼E的前後則分別決定出第一分斷點和第二分斷點。當移動程式碼E至新的位址範圍內，即出現移動後的程式碼E'。符號51和52分別表示插入的第一個/第二個無條件分支指令。當程式碼D執行完成後，即透過第一個無條件分支指令跳到程式碼E'；當程式碼E'執行完成後，則再透過第二個無條件分支指令跳回程式碼F。因此，原始程式碼的程式執行順序可以維持，僅需要多執行兩個無條件分支指令即可。

當完成步驟S15的連接動作後，CPU 1載入的前置動作已經大致完成，只剩下一般機器程式碼中常見的參考位址(reference address)尚未處理。如果在原始程式碼20中的參考位址與移動後的區段程式碼有關，則必須進行檢查和修正。因此，CPU 1必須修正相關的參考位址(S16)。第8圖表示在本實施例中修正參考位址的示意圖。如圖所示，原始程式碼20中有三種參考位址會與包含缺陷位址21的程式碼E有關，分別標示為R1、R2、R3。R1表示程式碼E



## 五、發明說明 (18)

中指令的參考位址指向程式碼E中位址的情況；R2表示其他程式碼(例如程式碼C)中指令的參考位址指向程式碼E中位址的情況；R3表示程式碼E中指令的參考位址指向其他程式碼(如程式碼B)的情況。這些參考位址必須在程式碼E移動後為程式碼E'後，修正其參考位址(如R1'，R2'，R3')，以便產生實際可以載入到記憶體3的機器程式碼60。

要修正參考位址，CPU 1可對於整個原始程式碼20進行再次掃描，以便找出與包含缺陷位址的區段程式碼相關的參考位址。另外，一般參考位址又可以細分為兩種定址模式(addressing mode)，分別為相對位址定址模式(relative addressing mode)和絕對位址定址模式(absolute addressing mode)。如果與上述三種參考位址情況一併考慮，則共有六種類型。在R1的參考位址中，只有絕對位址定址模式的參考位址才需要修正。在R3的參考位址中，只有相對位址定址模式的參考位址才需要修正。而在R2的參考位址中則相對及絕對位址都需要修正。當修正上述絕對位址定址模式的參考位址時(均是指向區段程式碼E)，則在此參考位址中加入程式碼E和移動後程式碼E'之間的相對位移量即可。若要修正上述相對位址定址模式的參考位址，亦可以利用程式碼E和移動後程式碼E'之間的相對位移量來達成。不過相對位址定址模式一般在使用上有其限制，亦即相對位址通常會限定在一定範圍內，因此在處理相對位址定址模式上會較為複雜。例如，本實



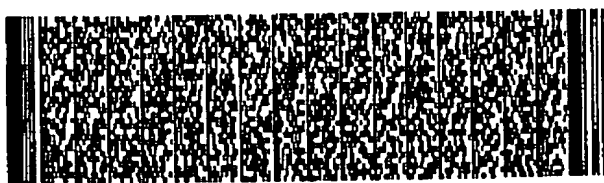
## 五、發明說明 (19)

施例中在缺陷位址21的前後都預留相當數量的可用記憶單元，則可以利用此部分插入連接的指令而間接達到相同的效果；另外，也可以選擇將缺陷位址21後面的所有程式碼往後方（即較高位址之方向）位移數個位元組，以便避開缺陷記憶單元，如此也可以避免因一區段程式碼移動過遠以致超過相對位址定址模式範圍的困擾；另一種解決方式則是調整編輯器的編碼格式，例如儘量採用絕對位址而避免使用相對位址進行編碼，或將因一區段程式碼移動過遠而受影響之相對位址之定址碼改為絕對位址定址碼。

當完成上述的前置處理，CPU 1的載入器便可以將原始程式碼、連接指令和移動後之區段程式碼載入到記憶體3中，完成所有的步驟(S17)。由於原本會佔用缺陷記憶單元的區段程式碼已經被移動，因此程式碼可以正常的執行和操作。

另外，上述說明雖然是以單一缺陷記憶單元為例，但是對於熟習此技藝者而言，可以延伸應用到複數個缺陷記憶單元的情況，其基本處理模式仍然相同。另外，本實施例中雖然係以位元組的掃描方式來決定出可分斷點，但對於熟習此技藝者而言，也可以將原始程式碼加以模組化(modularized)以方便決定可分斷點。此時不需要執行如本實施例中所描述的掃描動作，便可以利用較簡單的掃描程序決定出缺陷位址前後的第一分斷點和第二分斷點。

以上所揭露的局部缺陷記憶體之處理方法和系統，雖是以一般電腦系統為例，不過其最佳的應用範例應是在單



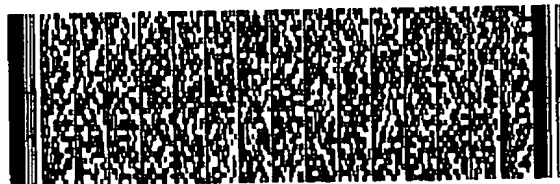
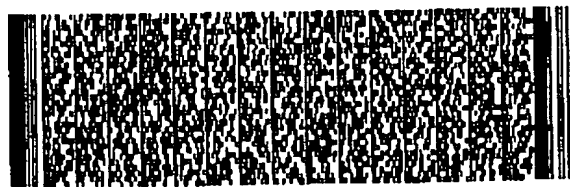
## 五、發明說明 (20)

晶片系統(system-on-a-chip)或單晶片電腦(single chip computer)上。在這類系統中，記憶體模組一般是內建的(embedded)，無法任意替換。在發現部分記憶單元有缺陷的情況下，便可以利用本實施例中所揭露的方法，整個晶片在處理後仍然可以正常執行程式碼。另外，本實施例中所揭露的局部缺陷記憶體亦可以適用在燒錄程式的應用上，例如快閃記憶體(flash memory)或是電子可抹式PROM(electrically erasable programmable read-only memory)的程式燒錄上，即使局部記憶單元有缺陷，仍然可以使用。最重要的是，本實施例在實施上並不會增加硬體設計的成本和複雜度，而對於軟體執行時的負擔也比習知技術來得輕。由此可知，本發明具有極高的產業利用價值。

## 第二實施例：

雖然第一實施例中已揭露一定程序的處理方式，可以解決局部缺陷記憶體的問題，但是其間仍有部分程序可以變更。例如，本實施例則是對於第一實施例改變其原始程式碼的掃描處理方式以及載入記憶體的順序而產生。

第9圖表示第二實施例之局部缺陷記憶體處理方法的流程圖。如第9圖所示，一方面CPU 1決定出缺陷記憶單元的實體位置(S20)並且決定出缺陷記憶單元在原始程式碼20中所對應的缺陷位址(S21)，另一方面也對於原始程式碼20進行掃描(S22)。由步驟S20和步驟S21可以決定出缺

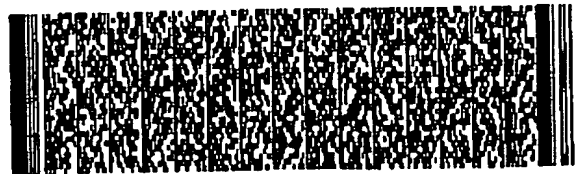
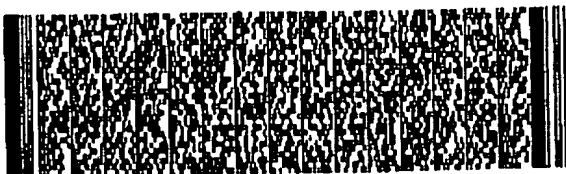


## 五、發明說明 (21)

陷記憶單元的缺陷位址，而由步驟S22則可以決定出原始程式碼20中的可分斷點。此同步處理可以適用於具有多工功能的CPU或是多重CPU的系統中。接著根據缺陷位址以及可分斷點的位址，便可以進一步決定出第一分斷點和第二分斷點(S23)，也就是需要移動的區段程式碼範圍。

接著的步驟順序則與第一實施例不同。先將整個原始程式碼20載入到記憶體3中(S24)，此時包含缺陷位址的區段程式碼亦會載入到缺陷記憶單元相鄰的記憶單元中，不過由於其中包含缺陷記憶單元，因此這部分的區段程式碼並不會實際使用。接著，再次載入此區段程式碼到記憶體3中的第一記憶單元和第二記憶單元之間(S25)，而在第一記憶單元和第二記憶單元之間則不包含缺陷記憶單元。接著則與第一實施例相同，插入連接指令以便連接區段程式碼和原始程式碼20中的其他部分(S26)，並且修正其中的參考位址(S27)。與第一實施例一樣，此時在記憶體3中的程式碼是可以正常執行的，並且實際執行的程式碼不會儲存到缺陷記憶單元內。

在第二實施例中，是先分別載入原始程式碼20和移動後的區段程式碼，再進行連接以及修正參考位址的處理。此樣的處理方式除了可以達到第一實施例相同的效果外，還可以得到較佳的處理效率。一般原始程式碼的來源大都是外部儲存媒介，例如硬碟機、軟碟機等等。要執行連接動作以及修正參考位址等處理時，在處理速度上較記憶體3來得慢，特別是在修正參考位址時，必須再次對於整個



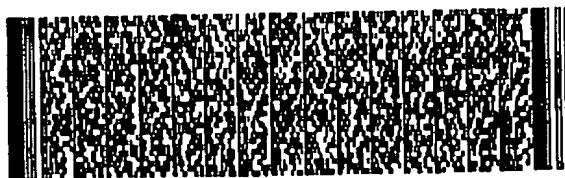
## 五、發明說明 (22)

原始程式碼3進行掃描。而在本實施例中，所有待處理的程式碼均已被載入於記憶體3中，因此可以更快地完成上述的處理步驟。另外，原始程式碼亦可儲存於一已知無缺陷之記憶體中，如ROM，RAM或PROM，可加快上述程式碼掃描之速度，不過此方式需要將該原始程式碼暫時或長久存放於另外之良好記憶體，或一記憶體中已確定不含缺陷記憶單元之部分。

## 第三實施例：

第一和第二實施例係適用於原始程式碼尚未載入記憶體中的情況，此時CPU(微處理器)可以在原始程式碼尚未載入記憶體前即施以前置處理，以便調整程式碼來避開缺陷記憶單元。而本實施例所要處理的情況，則是當原始程式碼已經成功地載入記憶體中，但是部分記憶單元在一段時間後卻顯示功能弱化的現象。這些功能弱化的記憶單元目前雖然可以正常讀取，但是在一定時間之後，從其讀出的邏輯值就會變得越來越難以辨認。此時必須對於記憶體內的原始程式碼內容加以調整，以避開這些功能弱化的缺陷記憶單元。

第10圖表示第三實施例之局部缺陷記憶體處理系統的架構圖。如第10圖所示，此時原始程式碼20已經載入於記憶體3中。而在系統操作過程中，則可以透過CPU 1本身或其他外部的處理器或測試器，周期性地對於記憶體3中的每個記憶單元進行測試。如果在測試中某個記憶單元所讀



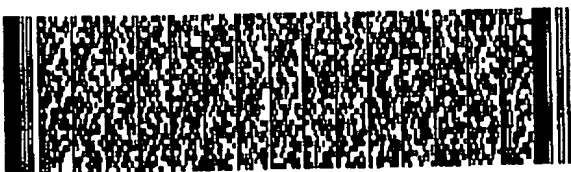
## 五、發明說明 (23)

出的資料已經越來越難以辨認時，則CPU 1即設定此記憶單元為缺陷記憶單元，並且執行本實施以下所揭露的處理方法，以便停止使用功能弱化的缺陷記憶單元。

第11圖表示第三實施例之局部缺陷記憶體處理方法的流程圖。如第11圖所示，原始程式碼20已經存在於記憶體3中(S30)。此時，一方面CPU 1可以對於原始程式碼20進行掃描(S31)，得到原始程式碼20中的所有可分斷點；另一方面也對於記憶體3進行周期性測試並且決定出其中功能弱化的缺陷記憶單元(S32)。由於原始程式碼20已經載入於記憶體3，因此可以根據缺陷記憶單元很快地判斷出對應的缺陷位址。接著根據缺陷位址以及所有可分斷點的位址，便可以進一步決定出第一分斷點和第二分斷點(S33)，也就是需要移動的區段程式碼範圍。

先在記憶體3中找出一段功能正常的記憶體區塊(即第一記憶單元和第二記憶單元之間)，以便容納需要移動的區段程式碼。接著將此區段程式碼複製到第一記憶單元和第二記憶單元之間(S34)。接著則與第一實施例相同，插入連接指令以便連接區段程式碼和原始程式碼20中的其他部分(S35)，並且修正其中的參考位址(S36)。此時在記憶體3中的程式碼是可以正常執行的，並且實際執行的程式碼不會儲存到功能弱化的缺陷記憶單元內。

綜合以上所述，本發明之局部缺陷記憶體的處理方法和系統具有下列優點：



## 五、發明說明 (24)

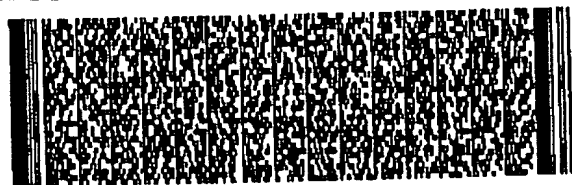
1. 本發明中解決局部缺陷記憶單元的方式，並不需  
要變更硬體(記憶體架構)上的設計或是增加硬體線路，而  
是透過修改程式碼的方式來避免使用到缺陷記憶單元；而  
程式碼在執行時所增加的負擔也不嚴重，在最佳情況下只  
需要加入兩個無條件分支命令(即JMP)即可。因此在實現  
複雜度以及成本上，本發明的確具有極佳的產業利用價  
值。

2. 本發明不僅可以適用於已知缺陷記憶體的情況，  
也可以適用於正在使用中的記憶體，如第三實施例所述。

習知技術中的硬體修正方式通常在執行時都必須將記  
憶體抽出單獨加以修改，而本發明則可以對於使用中的記  
憶體調整其中的程式碼，因此應用上更為方便。

另外，如上述記憶體中含複數個並不相鄰之缺陷記憶  
單元，亦可應用本發明產生數組與這些缺陷記憶單元相對  
應之可分斷點組，而將這些可分斷點組間之數個程式碼區  
段移至數個不同之新位址，並做相關之參考位址之更正，  
以產生一個在執行時可避開這些缺陷記憶單元之更改過的  
程式碼。

本發明雖以一較佳實施例揭露如上，然其並非用以限  
定本發明，任何熟習此項技藝者，在不脫離本發明之精神  
和範圍內，當可做些許的更動與潤飾，因此本發明之保護  
範圍當視後附之申請專利範圍所界定者為準。



## 六、申請專利範圍

1. 一種將程式碼載入記憶體以供執行的處理方法，適用於包含複數記憶單元之一記憶體和待載入於上述記憶體之一原始程式碼，其包括下列步驟：

決定上述記憶體是否包含缺陷記憶單元；

當上述記憶體不包含缺陷記憶單元時，則載入上述原始程式碼至上述記憶體；

當上述記憶體包含至少一缺陷記憶單元時，則執行下列步驟：

掃描上述原始程式碼，在上述缺陷記憶單元所對應於上述原始程式碼之一缺陷位址前後，決定第一分斷點和第二分斷點；

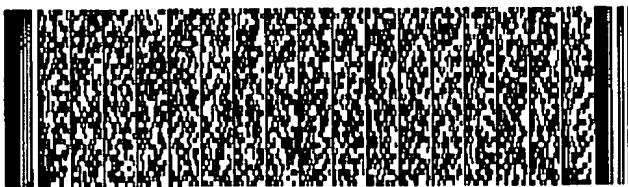
移動上述第一分斷點和上述第二分斷點之間的區段程式碼，至第一位址和第二位址之間，上述第一位址和上述第二位址之間之位址皆不對應於上述記憶體之上述缺陷位址；

當上述區段程式碼包含至少一可執行之指令時，連接移動後之上述區段程式碼與上述原始程式碼中未移動部分的執行順序；以及

載入上述原始程式碼中未移動部分、上述連接指令和移動後之上述區段程式碼至上述記憶體中。

2. 如申請專利範圍第1項所述之處理方法，其中尚包括一步驟：

修正上述區段程式碼與上述原始程式碼中未移動部分之間的參考位址。



## 六、申請專利範圍

3．如申請專利範圍第2項所述之處理方法，其中上述修正步驟係用以修正移動後之上述區段程式碼對於上述原始程式碼中未移動部分之相對定址模式參考位址，以及修正上述原始程式碼中未移動部分對於移動後之上述區段程式碼之相對定址模式參考位址以及絕對定址模式參考位址。

4．如申請專利範圍第2項所述之處理方法，其中尚包括一步驟：

修正移動後之上述區段程式碼內的絕對定址模式參考位址。

5．如申請專利範圍第1項所述之處理方法，其中掃描上述原始程式碼之步驟中更包括下列步驟：

依序讀取上述原始程式碼；

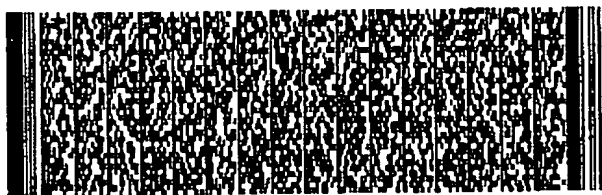
依據上述原始程式碼中各指令組成，輸出複數可分斷點；以及

根據上述缺陷位址和插入之上述連接指令，決定上述第一分斷點和上述第二分斷點。

6．如申請專利範圍第5項所述之處理方法，其中依序讀取上述原始程式碼之步驟中更包括下列步驟：

提供一第一資料表和一第二資料表，上述第一資料表用以記錄上述原始程式碼中條件分支指令之分支目的位址，上述第二資料表用以記錄已讀取的位址範圍；

當所讀取的指令為一條條件分支指令時，記錄上述條件分支指令之分支目的位址於上述第一資料表；



## 六、申請專利範圍

當完成讀取一指令時，更新上述第二資料表之位址範圍；以及

當所讀取的指令為一結束指令或其位址在上述第二資料表之位址範圍內時，並且當上述第一資料表之分支目的位址不屬於上述第二資料表之位址範圍時，則根據上述第一資料表之分支目的位址繼續讀取。

7．如申請專利範圍第1項所述之處理方法，其中上述連接步驟中，係插入第一無條件分支指令於上述第一分斷點之位址上，上述第一無條件分支指令之目的位址為上述區段程式碼移動後之第一位址，並且插入第二無條件分支指令於上述區段程式碼移動後之第二位址，上述第二無條件分支指令之目的位址為上述第二分斷點。

8．如申請專利範圍第1項所述之處理方法，其中上述第二分斷點係在上述原始程式碼之最後一個位元組之後。

9．一種記憶體處理系統，用以處理一包含複數記憶單元之記憶體，其包括：

一微處理器，耦接於上述記憶體，用以載入一原始程式碼，當上述原始程式碼被載入之記憶單元皆為良好記憶單元時，則載入上述原始程式碼於上述記憶體，當上述原始程式碼被載入之記憶單元包含至少一缺陷記憶單元時，則掃描上述原始程式碼，在上述缺陷記憶單元所對應於上述原始程式碼之一缺陷位址前後，決定第一分斷點和第二分斷點，並且移動上述第一分斷點和上述第二分斷點之間的區段程式碼至第一位址和第二位址之間，並且當上述區



## 六、申請專利範圍

段程式碼包含至少一可執行之指令時，連接上述區段程式碼與上述原始程式碼中未移動部分的執行順序，並且載入上述原始程式碼中未移動部分、上述連接指令和移動後之上述區段程式碼至上述記憶體中，其中上述第一位址和上述第二位址之間之程式碼所對應之記憶體位址不包含上述缺陷位址。

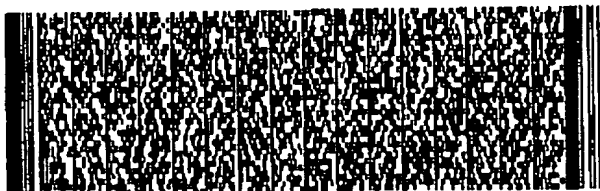
10．如申請專利範圍第9項所述之處理系統，其中上述第二分斷點係在上述原始程式碼之最後一個位元組之後。

11．如申請專利範圍第9項所述之處理系統，其中上述微處理器尚修正移動後之上述區段程式碼與上述原始程式碼中未移動部分之間的參考位址，以及移動後之上述區段程式碼內之參考位址。

12．如申請專利範圍第9項所述之處理系統，其中上述微處理器之連接動作中，係插入第一無條件分支指令於上述第一分斷點之位址上，上述第一無條件分支指令之目的位址為上述區段程式碼移動後之第一位址，並且插入第二無條件分支指令於上述區段程式碼移動後之第二位址，上述第二無條件分支指令之目的位址為上述第二分斷點。

13．如申請專利範圍第9項所述之處理系統，其中上述記憶體和上述微處理器係置於係置於同一晶片內。

14．如申請專利範圍第9項所述之處理系統，其中上述記憶體和上述微處理器係置於係置於獨立之不同晶片內。



## 六、申請專利範圍

15．一種防治局部弱化記憶體處理方法，適用於已載負一原始程式碼之一記憶體，其包括下列步驟：

檢查上述記憶體，用以找出上述記憶體中功能弱化之缺陷記憶單元；

當上述記憶體中包含至少一功能弱化之記憶單元，則執行下列步驟：

掃描上述原始程式碼，在上述缺陷記憶單元對應於上述原始程式碼之一缺陷位址前後，決定第一分斷點和第二分斷點；

移動上述第一分斷點和上述第二分斷點之間的區段程式碼，至上述記憶體中之第一記憶單元和第二記憶單元之間，上述第一記憶單元和上述第二記憶單元之間不包含上述缺陷記憶單元；以及

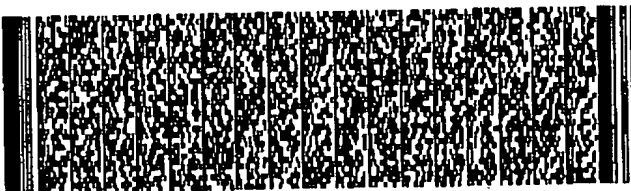
當上述區段程式碼包含至少一可執行之指令時，連接移動後之上述區段程式碼與上述原始程式碼中未移動部分的執行順序。

16．如申請專利範圍第15項所述之處理方法，其中上述第二分斷點係在上述原始程式碼之最後一個位元組之後。

17．如申請專利範圍第15項所述之處理方法，其中尚包括一步驟：

修正移動後之上述區段程式碼與上述原始程式碼中未移動部分之間的參考位址。

18．如申請專利範圍第17項所述之處理方法，其中上



## 六、申請專利範圍

述修正步驟係用以修正移動後之上述區段程式碼對於上述原始程式碼中未移動部分之相對定址模式參考位址，以及修正上述原始程式碼中未移動部分對於移動後之上述區段程式碼之相對定址模式參考位址以及絕對定址模式參考位址。

19．如申請專利範圍第18項所述之處理方法，其中尚包括一步驟：

修正移動後之上述區段程式碼內的絕對定址模式參考位址。

20．如申請專利範圍第15項所述之處理方法，其中掃描上述原始程式碼之步驟中更包括下列步驟：

依序讀取上述原始程式碼；

依據上述原始程式碼中各指令組成，輸出複數可分斷點；以及

根據上述缺陷位址和插入之上述連接指令，決定上述第一分斷點和上述第二分斷點。

21．如申請專利範圍第20項所述之處理方法，其中依序讀取上述原始程式碼之步驟中更包括下列步驟：

提供一第一資料表和一第二資料表，上述第一資料表用以記錄上述原始程式碼中條件分支指令之分支目的位址，上述第二資料表用以記錄已讀取的位址範圍；

當所讀取的指令為一條條件分支指令時，記錄上述條件分支指令之分支目的位址於上述第一資料表；

當完成讀取一指令時，更新上述第二資料表之位址範圍



## 六、申請專利範圍

圖；以及

當所讀取的指令為一結束指令或其位址在上述第二資料表之位址範圍內時，並且當上述第一資料表之分支目的位址不屬於上述第二資料表之位址範圍時，則根據上述第一資料表之分支目的位址繼續讀取。

22. 如申請專利範圍第15項所述之處理方法，其中上述連接步驟中，係插入第一無條件分支指令於上述第一分斷點之位址上，上述第一無條件分支指令之目的位址為上述區段程式碼移動後之第一記憶單元，並且插入第二無條件分支指令於上述區段程式碼移動後之第二記憶單元之後，上述第二無條件分支指令之目的位址為上述第二分斷點。

23. 一種記憶體處理系統，用以處理一包含複數記憶單元並且已儲存一原始程式碼之記憶體，其包括：

一微處理器，耦接於上述記憶體，用以檢查上述記憶體之記憶單元是否有功能弱化之記憶單元，當上述弱化之記憶單元存在並且已儲存上述原始程式碼時，則掃描上述原始程式碼，在上述弱化之記憶單元對應於上述原始程式碼之一弱化位址前後，決定第一分斷點和第二分斷點，並且移動上述第一分斷點和上述第二分斷點之間的區段程式碼至上述記憶體中之第一記憶單元和第二記憶單元之間，並且當上述區段程式碼包含至少一可執行之指令時，連接移動後之上述區段程式碼與上述原始程式碼中未移動部分的執行順序，其中上述第一記憶單元和上述第二記憶單元



## 六、申請專利範圍

之間不包含上述缺陷記憶單元。

24．如申請專利範圍第23項所述之處理系統，其中上述第二分斷點係在上述原始程式碼之最後一個位元組之後。

25．如申請專利範圍第23項所述之處理系統，其中上述微處理器係直接耦接於上述記憶體。

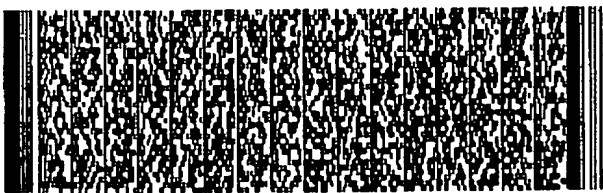
26．如申請專利範圍第23項所述之處理系統，其中上述微處理器尚修正移動後之上述區段程式碼與上述原始程式碼中未移動部分之間的參考位址，以及移動後之上述區段程式碼內之參考位址。

27．如申請專利範圍第23項所述之處理系統，其中上述微處理器之連接動作中，係插入第一無條件分支指令於上述第一分斷點之位址上，上述第一無條件分支指令之目的位址為上述區段程式碼移動後之第一記憶單元，並且插入第二無條件分支指令於上述區段程式碼移動後之第二記憶單元，上述第二無條件分支指令之目的位址為上述第二分斷點。

28．如申請專利範圍第23項所述之處理系統，其中上述記憶體和上述微處理器係置於係置於同一晶片內。

29．如申請專利範圍第23項所述之處理系統，其中上述記憶體和上述微處理器係置於係置於獨立之不同晶片內。

30．一種記憶體處理方法，適用於包含複數記憶單元之一記憶體和待載入於上述記憶體之一原始程式碼，其包



## 六、申請專利範圍

括下列步驟：

決定上述記憶體是否包含缺陷記憶單元；

當上述記憶體不包含缺陷記憶單元或上述原始程式碼不被載入於上述記憶體中缺陷記憶單元所對應之缺陷位址時，則載入上述原始程式碼至上述記憶體；

當上述原始程式碼被載入上述記憶體內至少一缺陷記憶單元所對應之缺陷位址時，則執行下列步驟：

掃描上述原始程式碼，在上述缺陷記憶單元所對應於上述原始程式碼之一缺陷位址前後，決定第一分斷點和第二分斷點；

載入上述原始程式碼至上述記憶體中；

載入上述第一分斷點和上述第二分斷點之間的區段程式碼，至第一記憶單元和第二記憶單元之間，上述第一記憶單元和上述第二記憶單元之間不包含上述缺陷記憶單元；以及

當上述區段程式碼包含至少一可執行之指令時，連接位於上述第一記憶單元和第二記憶單元之間的上述區段程式碼與上述原始程式碼中其他部分的執行順序。

31．如申請專利範圍第30項所述之處理方法，其中尚包括一步驟：

修正位於上述第一記憶單元和第二記憶單元之間的上述區段程式碼與上述原始程式碼中其他部分之間的參考位址。

32．如申請專利範圍第31項所述之處理方法，其中上



## 六、申請專利範圍

述修正步驟係用以修正位於上述第一記憶單元和第二記憶單元之間的上述區段程式碼對於上述原始程式碼中其他部分之相對定址模式參考位址，以及修正上述原始程式碼中其他部分對於位於上述第一記憶單元和第二記憶單元之間的上述區段程式碼之相對定址模式參考位址以及絕對定址模式參考位址。

33．如申請專利範圍第32項所述之處理方法，其中尚包括一步驟：

修正位於上述第一記憶單元和第二記憶單元之間的上述區段程式碼內的絕對定址模式參考位址。

34．如申請專利範圍第30項所述之處理方法，其中掃描上述原始程式碼之步驟中更包括下列步驟：

依序讀取上述原始程式碼；

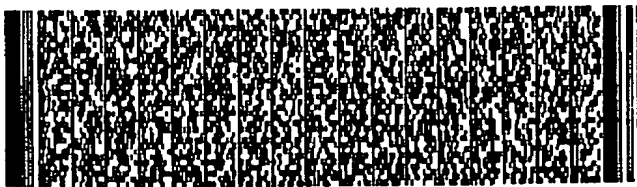
依據上述原始程式碼中各指令組成，輸出複數可分斷點；以及

根據上述缺陷位址和插入之上述連接指令，決定上述第一分斷點和上述第二分斷點。

35．如申請專利範圍第34項所述之處理方法，其中依序讀取上述原始程式碼之步驟中更包括下列步驟：

提供一第一資料表和一第二資料表，上述第一資料表用以記錄上述原始程式碼中條件分支指令之分支目的位址，上述第二資料表用以記錄已讀取的位址範圍；

當所讀取的指令為一條條件分支指令時，記錄上述條件分支指令之分支目的位址於上述第一資料表；



## 六、申請專利範圍

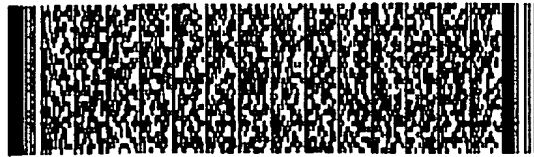
當完成讀取一指令時，更新上述第二資料表之位址範圍；以及

當所讀取的指令為一結束指令或其位址在上述第二資料表之位址範圍內時，並且當上述第一資料表之分支目的位址不屬於上述第二資料表之位址範圍時，則根據上述第一資料表之分支目的位址繼續讀取。

36．如申請專利範圍第30項所述之處理方法，其中上述連接步驟中，係插入第一無條件分支指令於上述第一分斷點之位址上，上述第一無條件分支指令之目的位址為上述區段程式碼之第一記憶單元，並且插入第二無條件分支指令於上述區段程式碼之第二記憶單元，上述第二無條件分支指令之目的位址為上述第二分斷點。

37．一種記憶體處理系統，用以處理一包含複數記憶單元之記憶體，其包括：

一微處理器，耦接於上述記憶體，用以載入一原始程式碼，當上述原始程式碼被載入之記憶單元包含至少一缺陷記憶單元時，則該系統能掃描上述原始程式碼，且在上述缺陷記憶單元所對應於上述原始程式碼之一缺陷位址前後，決定第一分斷點和第二分斷點，並且載入上述原始程式碼至上述記憶體中以及載入上述第一分斷點和上述第二分斷點之間的區段程式碼至第一記憶單元和第二記憶單元之間，並且當上述區段程式碼包含至少一可執行之指令時，連接位於上述第一記憶單元和第二記憶單元之間之上述區段程式碼與上述原始程式碼中其他部分的執行順序，



## 六、申請專利範圍

其中上述第一記憶單元和上述第二記憶單元之間不包含上述缺陷記憶單元。

38．如申請專利範圍第37項所述之處理系統，其中上述第二分斷點係在上述原始程式碼之最後一個位元組之後。

39．如申請專利範圍第37項所述之處理系統，其中上述微處理器尚修正位於上述第一記憶單元和第二記憶單元之間的上述區段程式碼與上述原始程式碼中其他部分之間的參考位址，以及位於上述第一記憶單元和第二記憶單元之間的上述區段程式碼內之參考位址。

40．如申請專利範圍第37項所述之處理系統，其中上述微處理器之上述連接動作中，係插入第一無條件分支指令於上述第一分斷點之位址上，上述第一無條件分支指令之目的位址為上述區段程式碼之第一記憶單元，並且插入第二無條件分支指令於上述區段程式碼之第二記憶單元，上述第二無條件分支指令之目的位址為上述第二分斷點。

41．如申請專利範圍第37項所述之處理系統，其中上述記憶體和上述微處理器係置於係置於同一晶片內。

42．如申請專利範圍第37項所述之處理系統，其中上述記憶體和上述微處理器係置於係置於獨立之不同晶片內。

43．一種程式碼掃描處理方法，用以決定一原始程式碼之可分斷點，上述程式碼掃描處理方法包括下列步驟：

提供一第一資料表和一第二資料表，上述第一資料表



## 六、申請專利範圍

用以記錄上述原始程式碼中的條件分支指令相關之待掃描位址，上述第二資料表用以記錄已讀取的位址範圍；

依序讀取上述原始程式碼；

當所讀取的指令為一條條件分支指令時，記錄上述條件分支指令相關之待掃描位址於上述第一資料表，並且依上述條件分支指令之下一位址或分支目的位址繼續掃描；

隨指令之讀取，更新上述第二資料表之位址範圍，並且當完成讀取至少一完整指令時，輸出可分斷點；以及

當所讀取的指令為一結束指令或其位址在上述第二資料表之位址範圍內時，則根據上述第一資料表之一待掃描位址繼續讀取，並將該待掃描位址從上述第一資料表中除去。

44．如申請專利範圍第43項所述之處理方法，其中上述程式碼掃描處理方法係在一原始程式碼載入一記憶體時，決定可移動之區段程式碼。

45．如申請專利範圍第44項所述之處理方法，其中上述記憶體包含至少一缺陷記憶單元。

46．如申請專利範圍第43項所述之處理方法，其中上述條件分支指令相關之待掃描位址為上述條件分支指令之分支目的位址。

47．如申請專利範圍第43項所述之處理方法，其中上述條件分支指令相關之待掃描位址為上述條件分支指令之下一位址。

48．如申請專利範圍第43項所述之處理方法，其中尚



## 六、申請專利範圍

## 包括一步驟：

當所讀取的指令為一無條件分支指令時，則由上述無條件分支指令之分支目的位址繼續掃描。

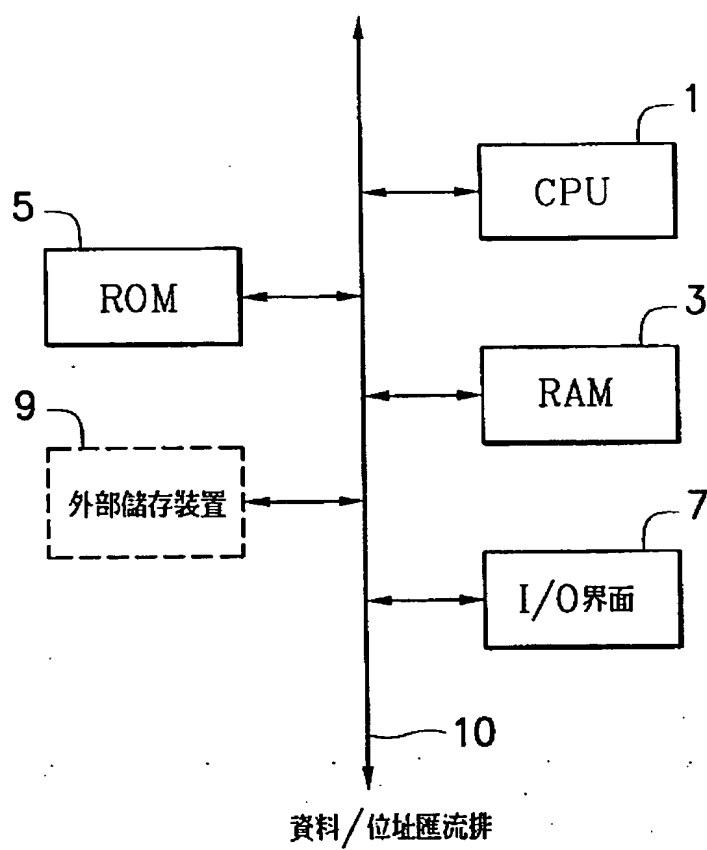
49．如申請專利範圍第43項所述之處理方法，其中當上述所讀取的指令為一結束指令或其位址在上述第二資料表之位址範圍內，且上述第一資料表中無待掃描位址時，則結束掃描。

50．如申請專利範圍第43項所述之處理方法，其中當掃描結束時，將仍未被掃描到之一連續程式碼位址判定為一資料程式碼區段。

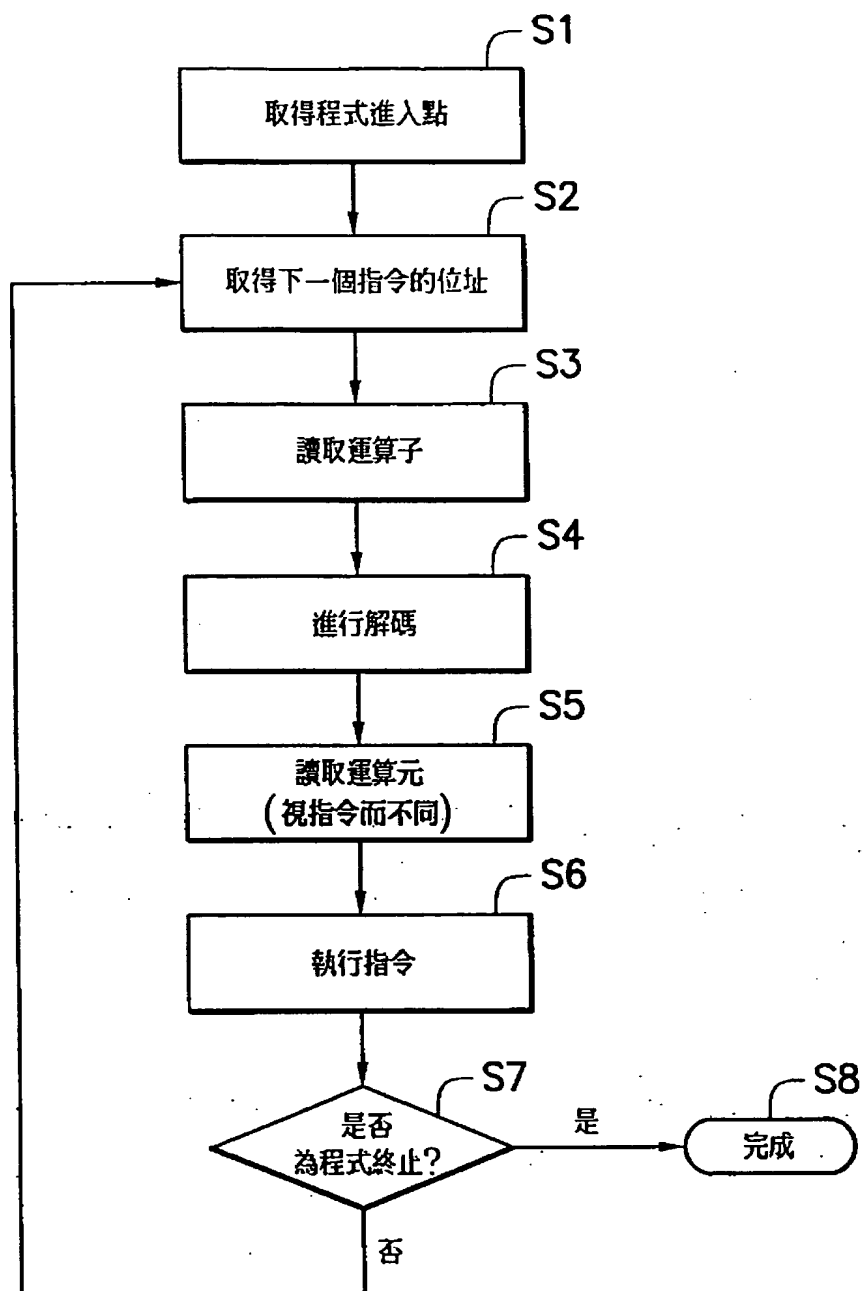
51．如申請專利範圍第43項所述之處理方法，其中當掃描結束時，將仍未被掃描到之一連續程式碼位址判定為一資料或堆疊程式碼區段。

52．如申請專利範圍第43項所述之處理方法，其中當掃描結束時，將仍未被掃描到之一連續程式碼位址判定為一非指令區段。

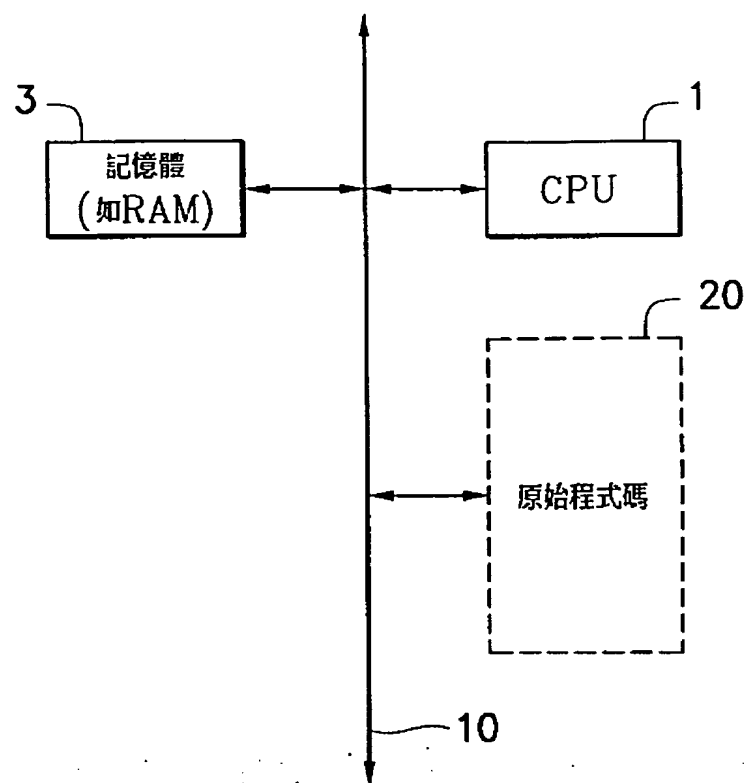




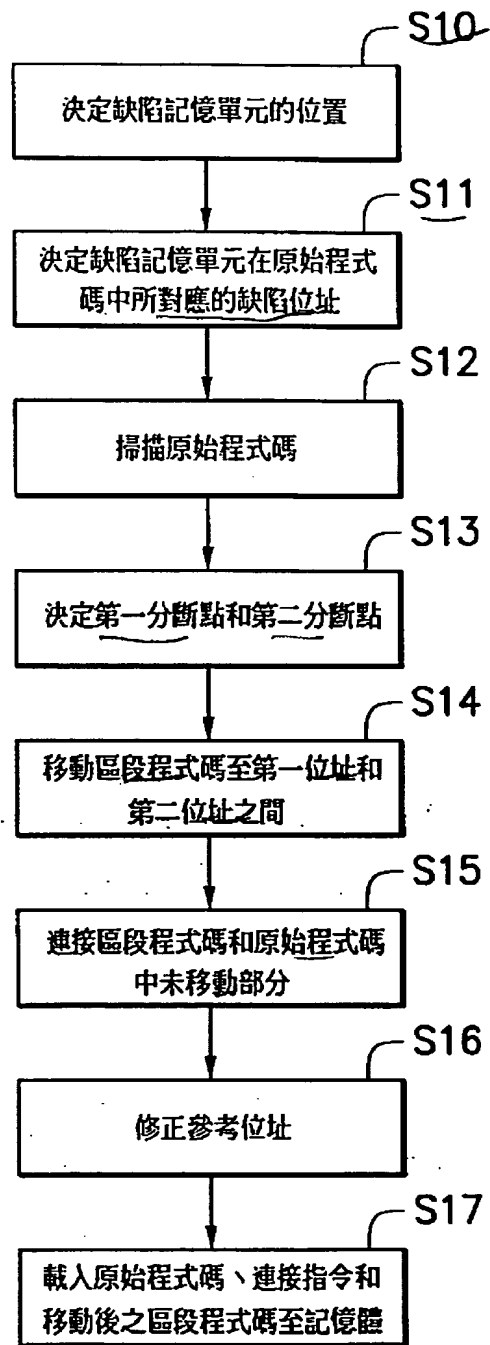
第 1 圖



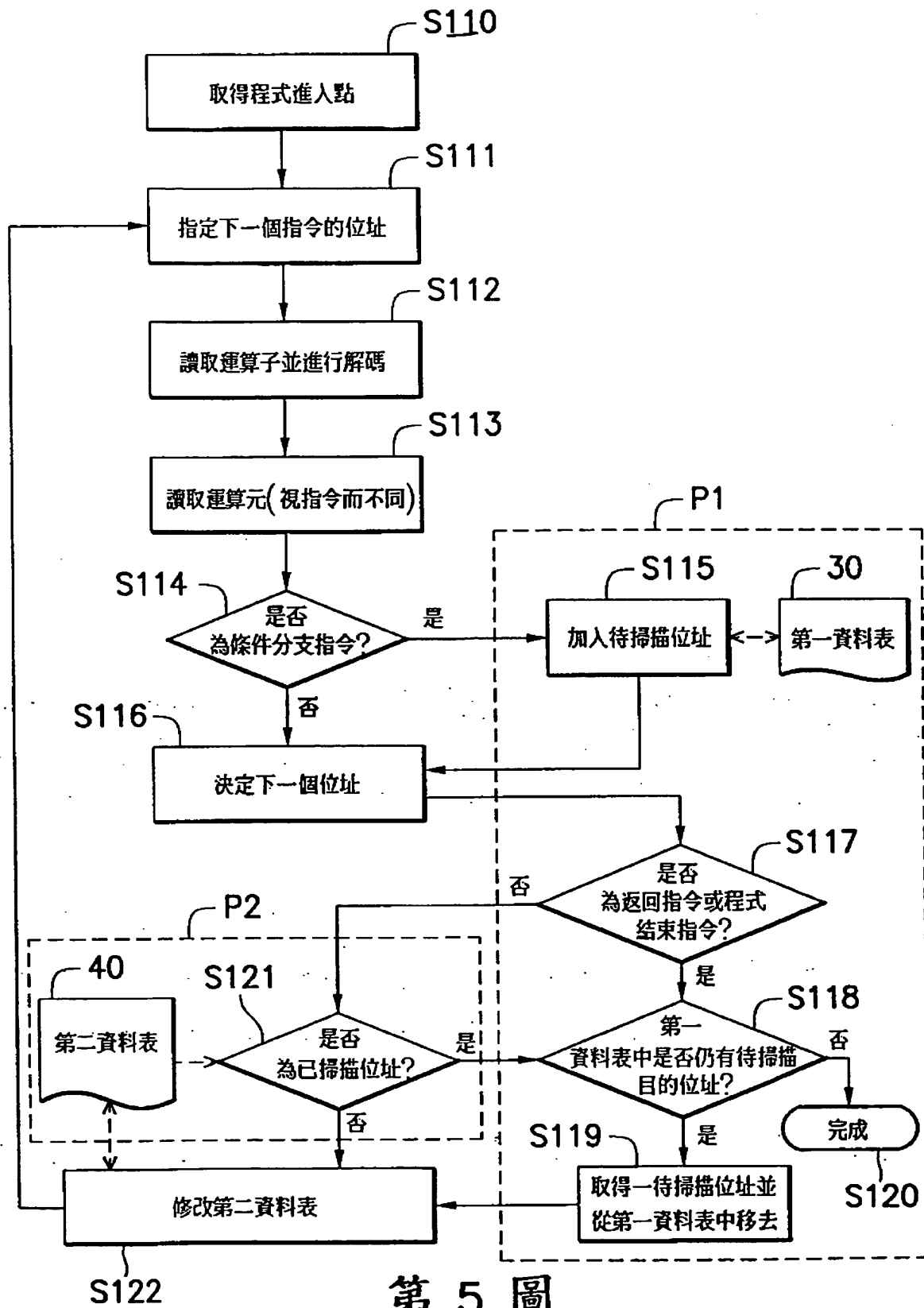
第 2 圖

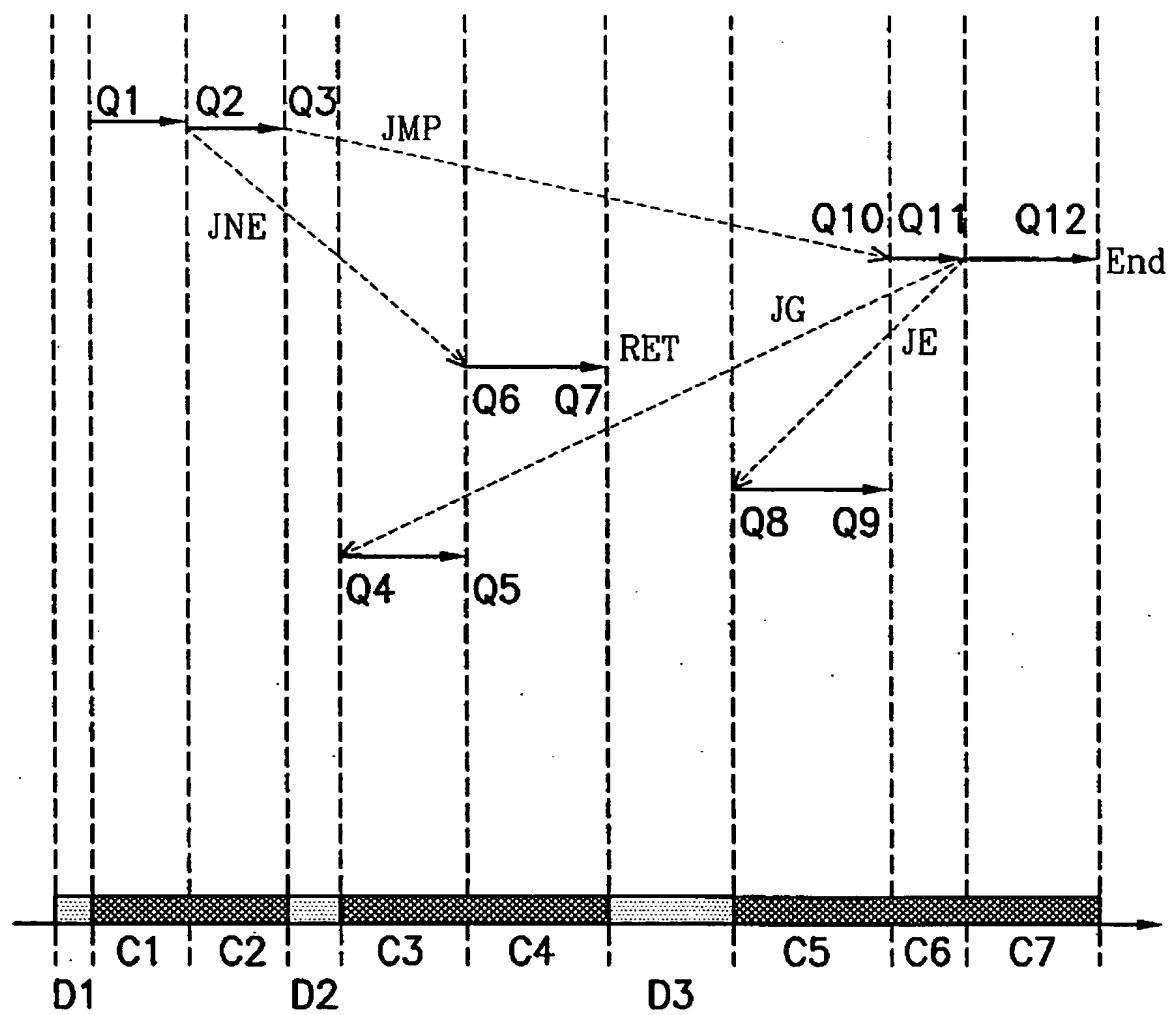


第 3 圖

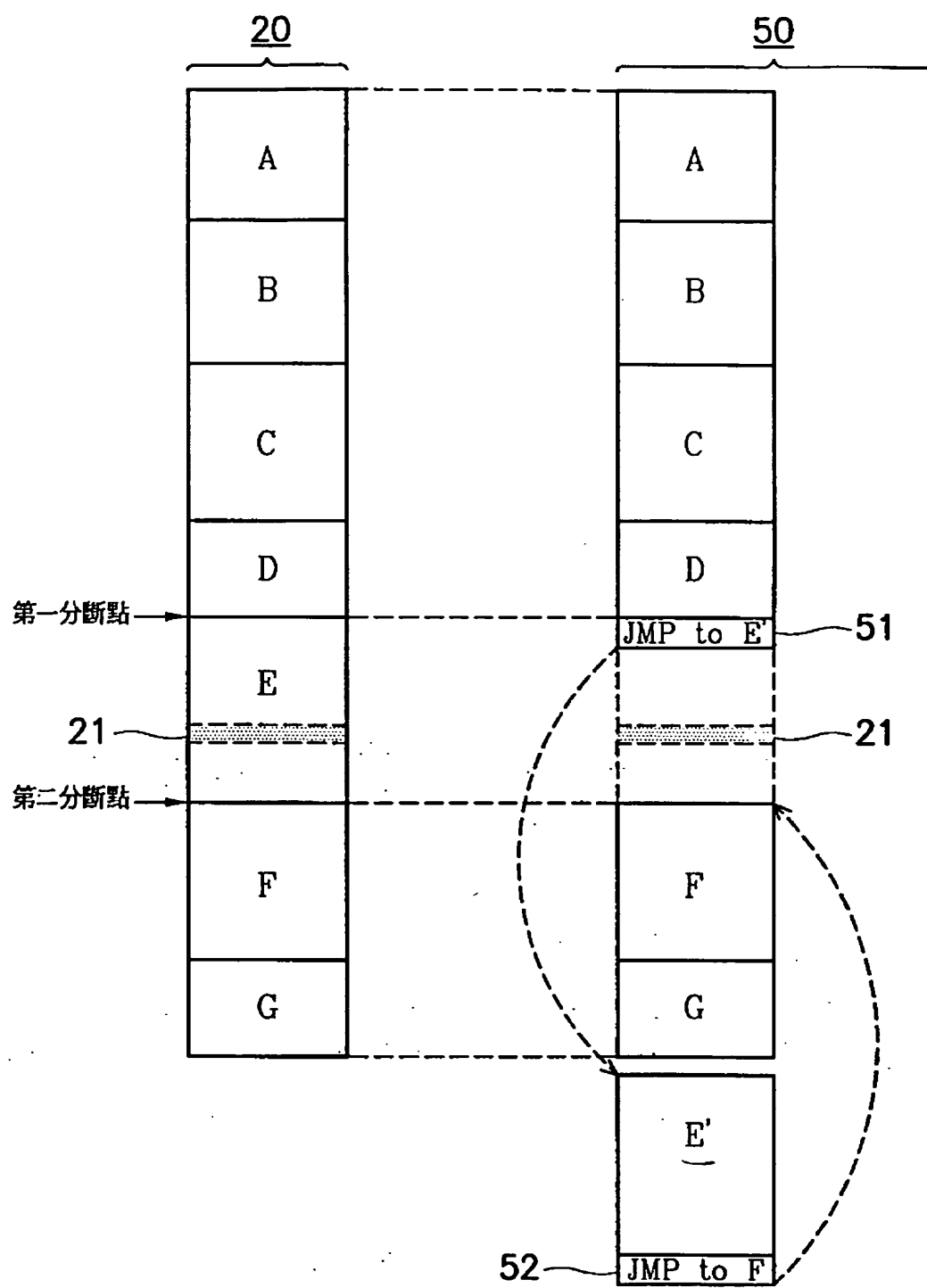


第 4 圖

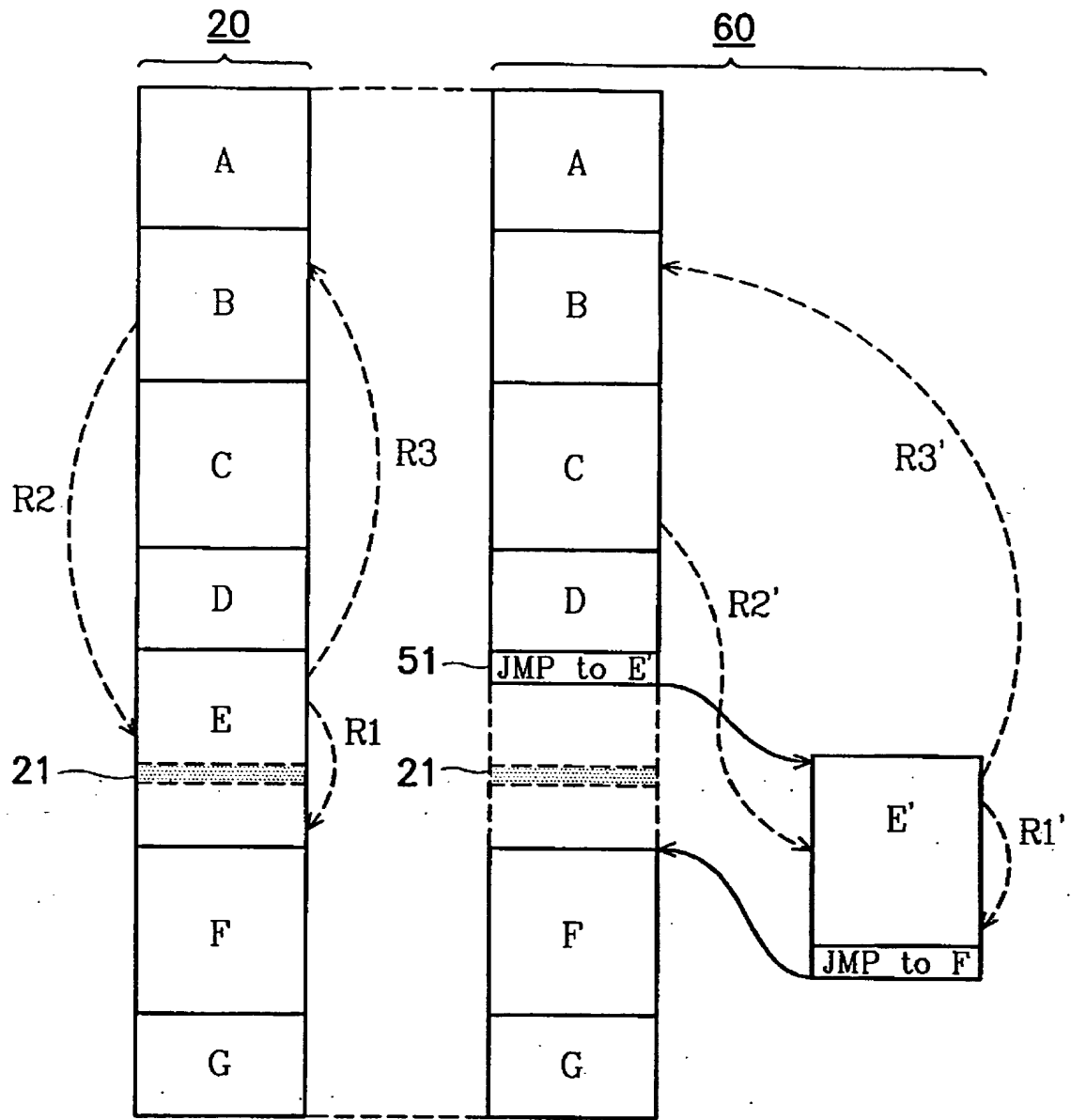




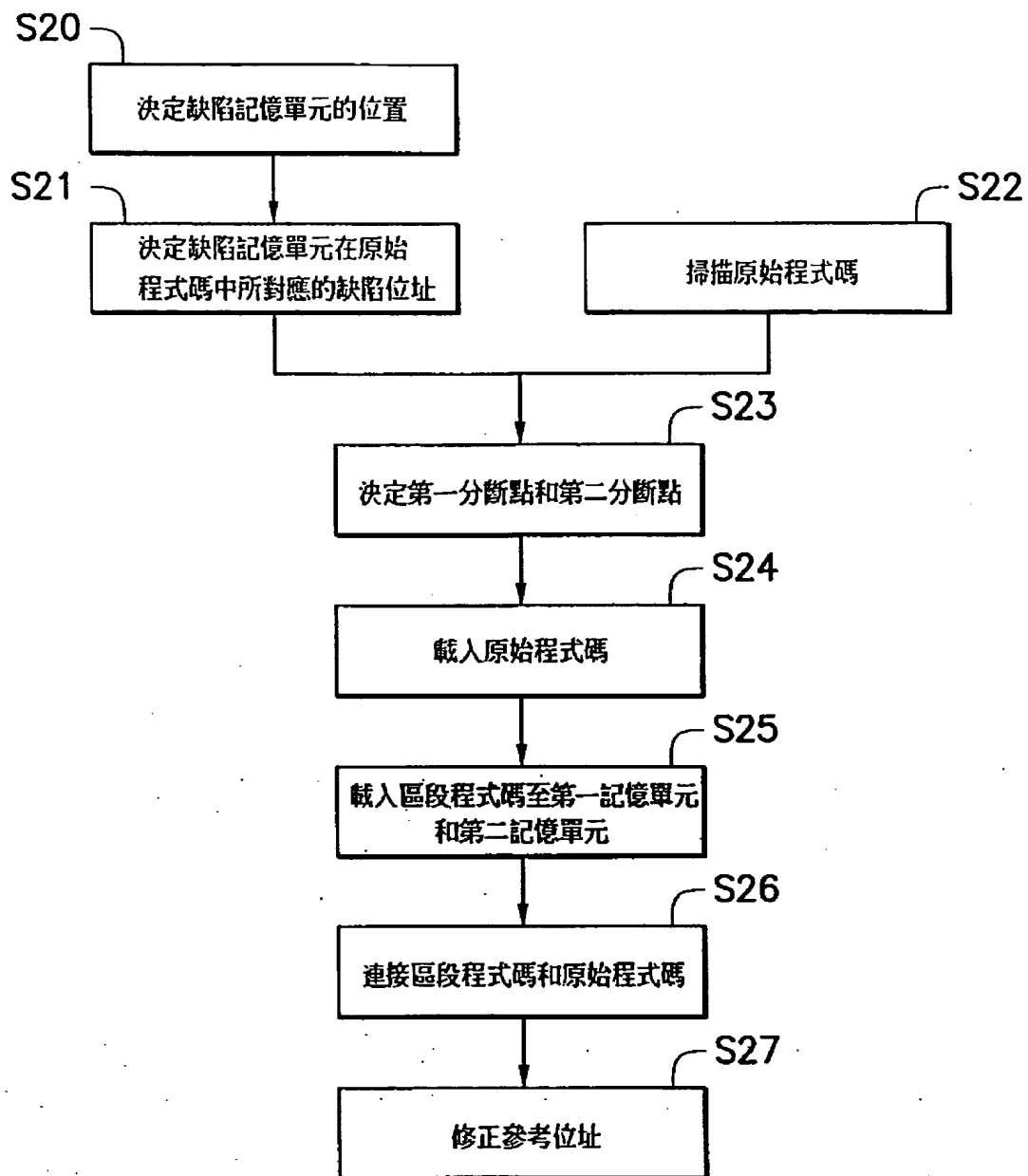
第 6 圖



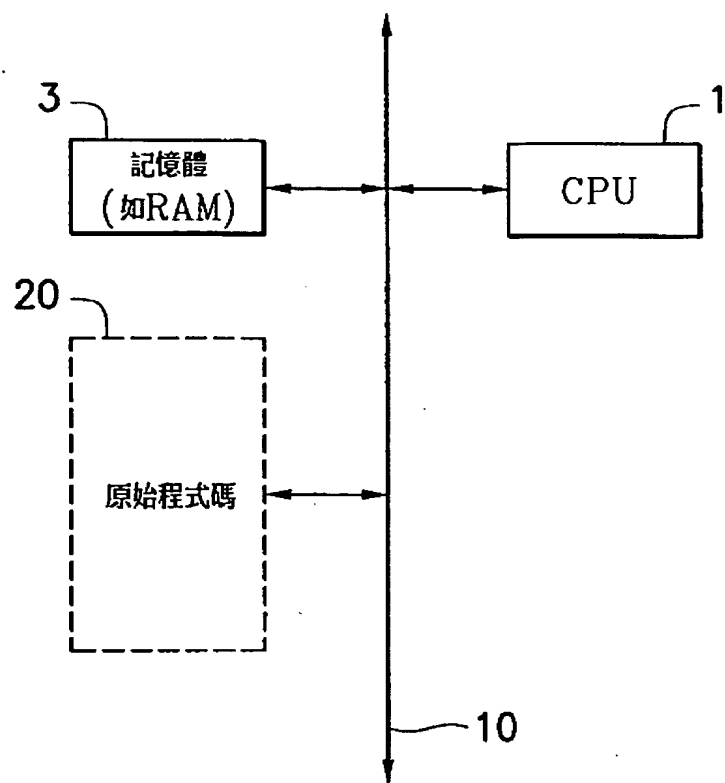
第 7 圖



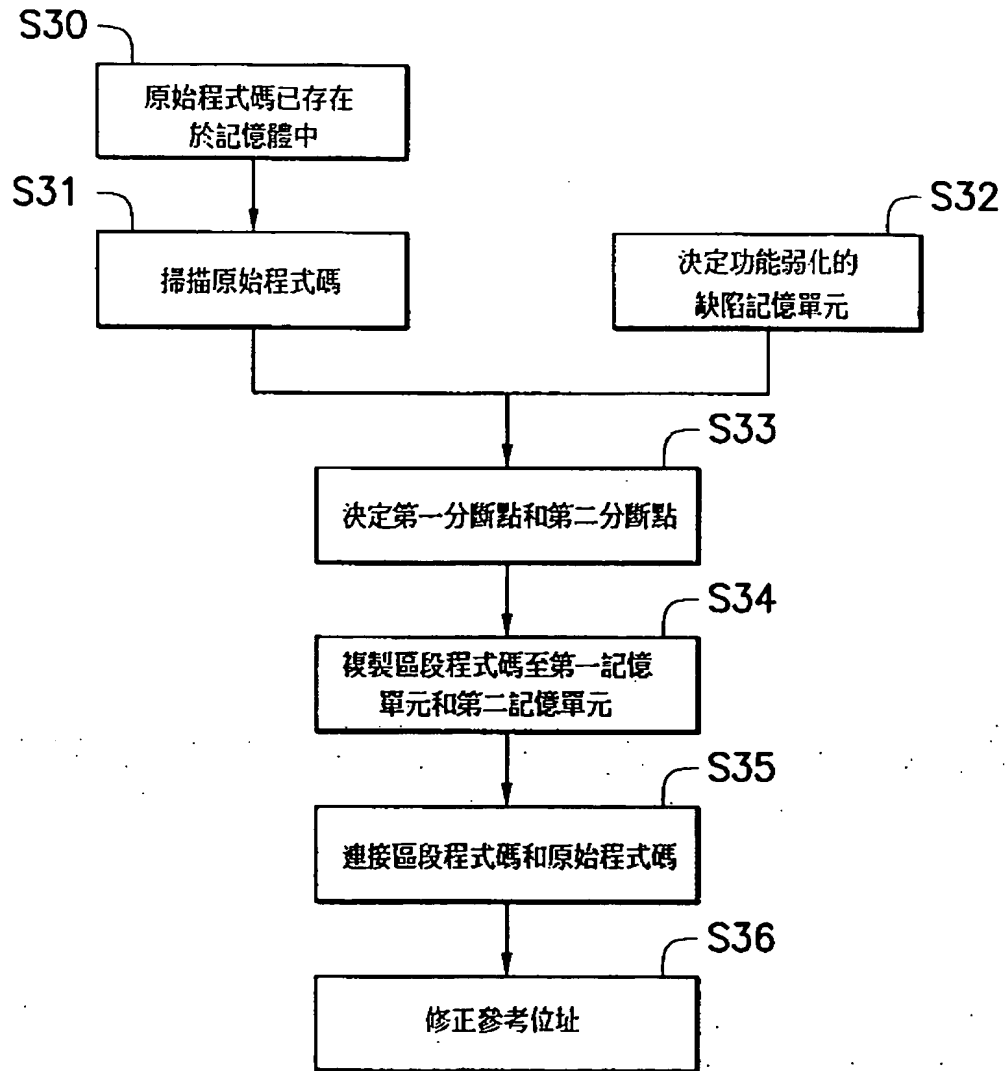
第 8 圖



第 9 圖



第 10 圖



第11圖